AMATH 482 Homework 2

Wenxuan Liu

February 10, 2021

Abstract

We are given two music files that play clips of the songs Sweet Child O' Mine by Guns N' Roses (GNR) and Comfortably Numb by Pink Floyd (Floyd), respectively. The goal is find the music score for guitar in GNR and bass & guitar in Floyd. To process GNR, we use Gabor Transform to get the spectrogram. Then we use Shannon Filter to get rid of overtones with high frequencies. To process bass in Floyd, we first Fourier transform the signal to frequency space and use Shannon Filter to keep only the low-frequency part and transform it back to time domain. Then we use Gabor Transform to get the spectrogram of low frequencies. To process guitar in Floyd, we do the same process but use different Shannon Filter (keep medium frequencies).

1 Introduction and Overview

We are given two music files that play clips of the songs Sweet Child O' Mine by Guns N' Roses and Comfortably Numb by Pink Floyd, respectively. The goal is find the music score for guitar in GNR and bass & guitar in Floyd. In order to get the music score, we need not only what frequencies are being played, but also at what time it is played. We need information from both the frequency domain and time domain, so instead of using Fourier Transformation, we use Gabor Transformation to get the spectrogram of both songs.

For GNR, we use Gabor Transform to get the spectrogram. Then we find out that we also have a lot of overtones with high frequencies, so use a Shannon Filter to filter out overtones with high frequencies. To get music score for bass and guitar in Floyd, we first Fourier transform the signal to frequency space and use Shannon Filter to keep only the frequencies we need and transform it back to time domain. We use two different Shannon Filters to get isolate the guitar and bass in Floyd. Then we use Gabor Transform to get the spectrogram. Know the corresponding music score for each frequency, we can convert the spectrograms to music scores.

2 Theoretical Background

2.1 Fourier Transformation

As we learned from our text book (lecture 1) [1], Fourier introduced the concept of representing a given function $f(x)$ by a trigonometric series of sines and cosines:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_n \cos kx + b_n \sin kx) \quad x \in [-\pi, \pi]. \tag{1}$$

Based on this idea, we have Fourier transformation that can bring any function from time/spacial domain to frequency domain:

$$\hat{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ikx} dx \tag{2}$$

The inverse Fourier transformation is the inverse transformation of Fourier transformation, which brings the function back from frequency domain:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(k) e^{ikx} dk \tag{3}$$

For a sequence of N values, we have discrete Fourier Transform:

$$\hat{x} = \frac{1}{N}\sum_{n=0}^{N-1} x_n e^{\frac{2\pi i k n}{N}} \tag{4}$$

Fast Fourier Transform is an algorithm that run in O(N log(N)). This algorithm allow us to do discrete Fourier Transform (Equation 4) fast. MATLAB has built-in function that perform FFT on data sets.

2.2 Gaussian and Shannon filter

From text book (lecture 2) [1], we learned Gaussian filters, which has the form:

$$F(k) = e^{-\tau(k-k_0)^2} \tag{5}$$

From text book (lecture 9) [1], we learned Shannon filters, which has the form:

$$F(k) = \begin{cases} 1 \text{ (a<k<b)} \\ 0 \text{ otherwise} \end{cases} \tag{6}$$

By applying filter in frequency space, we can preserve the frequency we want and git rid of the noisy data. After transform the filtered frequency data back to time/spacial domain, the data is less noisy.

2.3 Gabor Transform

From text book (lecture 4&7) [1], we learned Gabor transform, or the short-time Fourier transform (STFT), which has the form:

$$\widetilde{f}_g(\tau, k) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-ikx}dt \tag{7}$$

The Gabor Transformation give us information of the frequencies near some time τ, so if we change τ, we can get frequency information across the time domain. To apply Gabor Transformation on our data, we need the discrete form of it:

$$\widetilde{f}_g(m, n) = \int_{-\infty}^{\infty} f(t)g(t-nt_0)e^{2\pi i m\omega_0 t}dt \tag{8}$$

Gabor Transformation will allow us to create spectrograms for our data.

3 Algorithm Implementation and Development

We start by loading the songs: 10s GNR clip and 60s Floyd clip. We process the two songs in two sections below:

3.1 GNR guitar music score

1. First we prepare/interpret the data and define some helper variables. Since FFT takes spacial data in domain $[-\pi, \pi]$ and we want frequency in Hz, we scale the time data by $2\pi\frac{2\pi}{L}$, where L =length of GNR in seconds. We let n to be the number of data points in GNR; we define k for the frequency domain. Lastly we need a window width for Gabor transformation and step length of time domain. We pick window size $a = 1000$ and $step = 0.1s$ and create τ corresponding to step.

2. Next, we create Shannon filter to exclude all frequencies greater than 1000Hz. Then use Gabor Transformation to get the spectrogram data. See Algorithm 1 for sudo-code.

Algorithm 1: Gabor Transform Algorithm

Define Shannon Filter
for $j = 1 : length(\tau)$ **do**
  Create Gaussian filter centered at τ_j
  Multiply the data with the Gabor Window (Gaussian Filter)
  Fourier transform the filtered data and multiply with Shannon Filter
  Shift the frequency data and store in spectrum
end for

3. In the end, we plot the spectrogram and compare the frequencies of the bright points with the chart in Appendix C to get the actual music score.

3.2 Floyd Bass music score

1. First we prepare/interpret the data and define some helper variables exactly same as what we did for GNR, except this time with different window width for Gabor transformation and step length of time domain. We pick window size $a = 2000$ and $step = 1s$ and create τ corresponding to step.

2. Next, we want to isolate the bass frequency. We perform Fourier Transformation on the data, then create Shannon filter to exclude all frequencies greater than 200Hz. Then we can transform the frequency data back to time domain. Now our clip has only low-frequency component. See Algorithm 2 for sudo-code.

Algorithm 2: Shannon Filtering Algorithm

yt = fft(y);
Create Shannon Filter only preserve 0-200Hz frequency data
Multiply shifted frequency data with shifted filter
Transform data back to time domain for next step

3. We Gabor Transformation to get the spectrogram data and use Gaussian filter to make the plot look cleaner. See Algorithm 3 for sudo-code.

Algorithm 3: Gabor Transform with Gaussian Algorithm

Define Shannon Filter
for $j = 1 : length(\tau)$ **do**
  Create Gaussian filter centered at τ_j
  Multiply the data with the Gabor Window (Gaussian Filter)
  Fourier transform the filtered data
  Create another Gaussian filter centered at maximum
  Filter data again with second Gaussian filter and store into spectrum
end for

4. In the end, we plot the spectrogram and compare the frequencies of the bright points with the chart in Appendix C to get the actual music score.

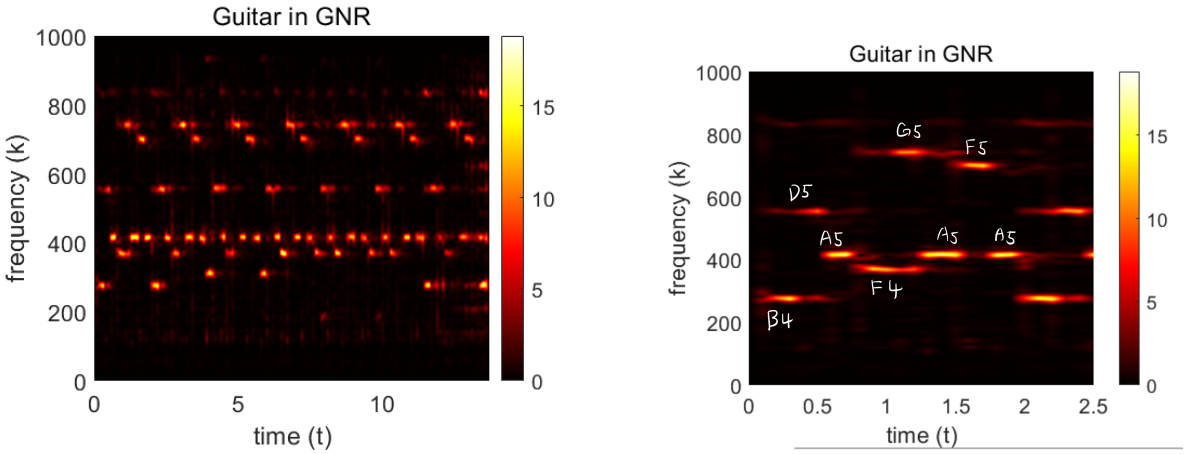
3.3 Floyd Guitar music score

For this section, we do the exact same process as 3.2, except we use a Shannon Filter from 300Hz to 1200Hz. The spectrogram over the 60 seconds time domain does not give us specific enough frequency information, so we zoom into first 0-10 seconds of the song. Do Algorithm 3 on this smaller piece of song, and get the resulting spectrogram. We repeat this for the rest of the song.

4 Computational Results

4.1 GNR guitar music score

Through the first picture, we can see a clear pattern: there are about 7 repetitions of same notes. We look into just the first copy of the music score, and label the scores in it. See Figure 1 for the whole spectrum and labeled music score. (The label A_5 has meaning the 5^{th} A from left of Appendix C).



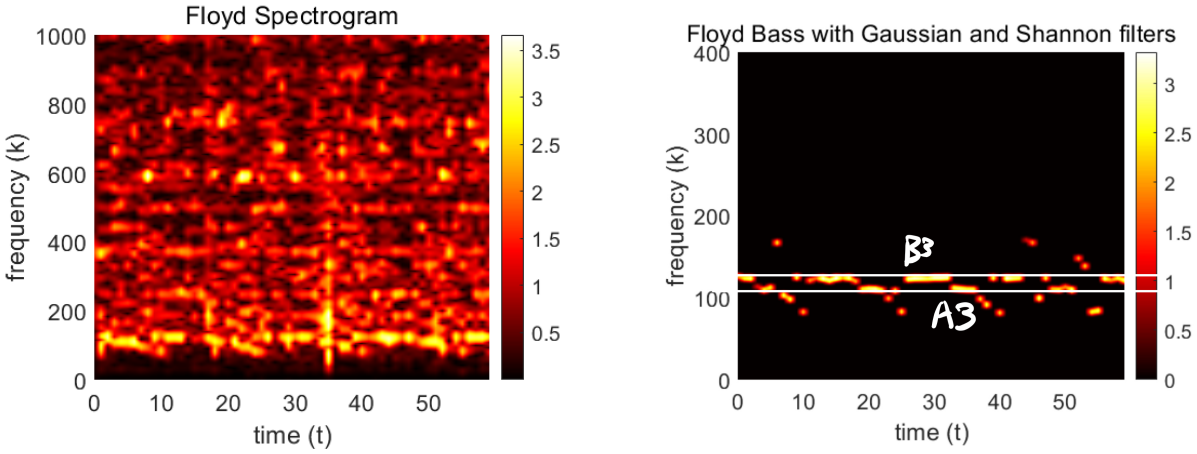
(a) GNR Spectrogram

(b) Labeled Repetition

Figure 1: The spectrum and one repetition being labeled

4.2 Floyd Bass music score

We first obtain a spectrogram without any filter, and we can see an obvious pattern in the low frequency portion. We isolate the low frequency and maximum points with filters to get a clearer result. See Figure 2 for the whole spectrum and isolated bass music score. (The label A_5 has meaning the 5^{th} A from left of Appendix C).



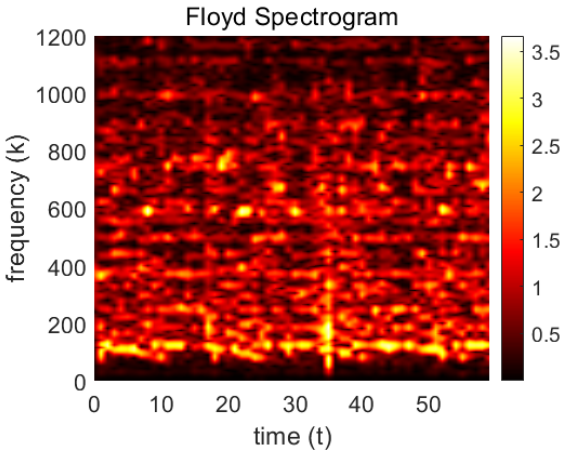
(a) Floyd Spectrogram

(b) Labeled Bass Score

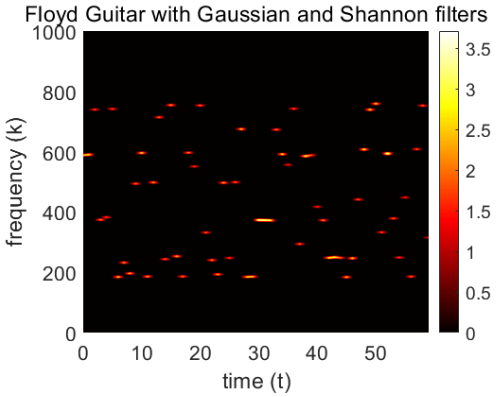
Figure 2: The spectrum and one repetition being labeled

4.3 Floyd Guitar music score

We first obtain a spectrogram without any filter, and we see some bright spots in medium frequency (300-1200). We isolate the medium frequency and maximum points with filters to get a clearer result. See Figure 3 for the whole spectrum and isolated bass music score. (The label A_5 has meaning the 5^{th} A from left of Appendix C).



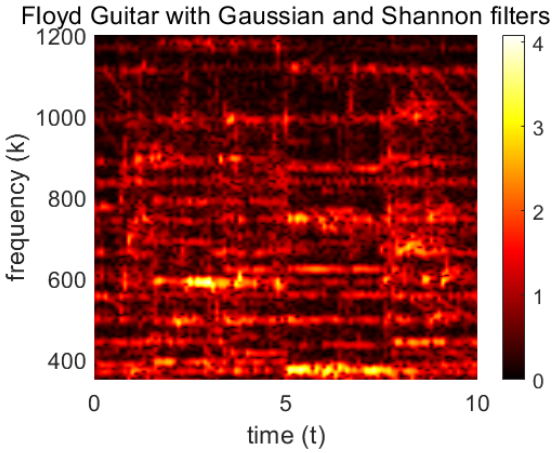
(a) Floyd Spectrogram



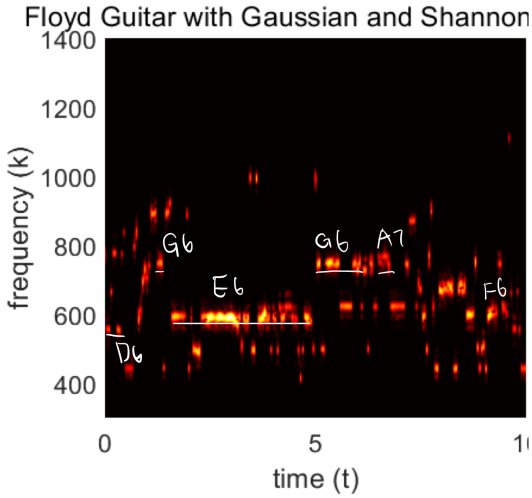
(b) Larger scale Guitar Score

Figure 3: The spectrum and one repetition being labeled

Then we create spectrogram only for the first 10 seconds of the song to get a closer look a the music score. See Figure 4 for zoom in spectrogram and labeled music score.



(a) Zoom-In Spectrogram



(b) Zoom-In Spectrogram labeled

Figure 4: The spectrum and one repetition being labeled

The 20-50s of the song is divided into 5 pieces, processes and results are similar.

5 Summary and Conclusions

After a series of computing, we can get a clear music score for the guitar played in GNR and bass played in Floyd. However, the guitar music score in Floyd is harder to recognize compare to guitar in GNR. This is probably because of the noise introduced by the other music instruments, which suggests that we need better strategies to isolate guitar from a lot of other music instruments.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

Appendix A MATLAB Functions

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `Y = fft(X)` computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm.
- `Y = fftshift(X)` rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.
- `X = ifft(Y)` computes the inverse discrete Fourier transform of Y using a fast Fourier transform algorithm. X is the same size as Y.
- `[M,I] = max(A)` also returns the index into the operating dimension that corresponds to the maximum value of A for any of the previous syntaxes.
- `pcolor(X,Y,C)` specifies the x- and y-coordinates for the vertices. The size of C must match the size of the x-y coordinate grid. For example, if X and Y define an m-by-n grid, then C must be an m-by-n matrix.
- `[y,Fs] = audioread(filename)` reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.

Appendix B MATLAB Code

MATLAB code that produce the outcomes discussed in this paper.

```
% Clean workspace
clear all; close all; clc
[y, Fs] = audioread('GNR.m4a');
tr_gnr = length(y)/Fs; % record time in seconds
L = tr_gnr; n = length(y);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);
t2 = linspace(0,L,n+1); t = t2(1:n);

a = 1000;
tau = 0:0.1:tr_gnr;

filter = zeros(size(y));
filter([1:13000+1], 1) = ones(13001, 1);
for j = 1:length(tau)
   g = exp(-a*(t - tau(j)).^2); % Window function
   yg = g.*transpose(y);
```

```
   ygt = fft(yg) .* transpose(filter);
   ygt_spec(:,j) = fftshift(abs(ygt));
end

figure(1)
pcolor(tau,ks,ygt_spec)
shading interp
set(gca,'ylim',[0, 1000],'xlim',[0, 2.5],'Fontsize',16)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title('Guitar in GNR');
%%
clear all; close all; clc

[y, Fs] = audioread('Floyd.m4a');
y = y(1:length(y)-1);
tr_gnr = length(y)/Fs; % record time in seconds
L = tr_gnr; n = length(y);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);
t2 = linspace(0,L,n+1); t = t2(1:n);

a = 2000;
tau = 0:1:tr_gnr;
for j = 1:length(tau)
   g = exp(-a*(t - tau(j)).^2); % Window function
   yg = g.*transpose(y);
   ygt = fftshift(abs(fft(yg)));
   ygt_spec(:,j) = ygt;
end

figure(1)
pcolor(tau,ks,log(ygt_spec+1))
shading interp
set(gca,'ylim',[0, 1200],'Fontsize',16)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title('Floyd Spectrogram');
%%
clear all; close all; clc

[y, Fs] = audioread('Floyd.m4a');
y = y(1:length(y)-1);
tr_gnr = length(y)/Fs; % record time in seconds
L = tr_gnr; n = length(y);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);
t2 = linspace(0,L,n+1); t = t2(1:n);

a = 2000;
tau = 0:1:tr_gnr;

yt = fft(y);
filter = zeros(size(y));
filter([1:175*60+1], 1) = ones(175*60+1, 1);
ytf = fftshift(yt) .* fftshift(filter);
y2 = fftshift(ifft(ifftshift(ytf)));
```

```
for j = 1:length(tau)
   g = exp(-a*(t - tau(j)).^2); % Window function
   yg = g.*transpose(y2);
   ygt = fftshift(abs(fft(yg)));
   [M, I] = max(ygt);
   g2 = exp(-0.1*(k - k(I)).^2);
   ygt_spec(:,j) = ygt .* g2;
end

figure(1)
pcolor(tau,ks,log(ygt_spec+1))
shading interp
set(gca,'ylim',[0, 400],'Fontsize',16)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title('Floyd Bass with Gaussian and Shannon filters');
%%

clear all; close all; clc

[y, Fs] = audioread('0-10.m4a');
%y = y(1:length(y)-1);
tr_gnr = length(y)/Fs; % record time in seconds
L = tr_gnr; n = length(y);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);
t2 = linspace(0,L,n+1); t = t2(1:n);

a = 800;
tau = 0:0.1:tr_gnr;

yt = fft(y);
filter = zeros(size(y));
filter([300*10: 1200*10], 1) = ones((1200-300)*10+1, 1);
ytf = fftshift(yt) .* fftshift(filter);
y2 = fftshift(ifft(ifftshift(ytf)));
for j = 1:length(tau)
   g = exp(-a*(t - tau(j)).^2); % Window function
   yg = g.*transpose(y2);
   ygt = fftshift(abs(fft(yg)));
   %[M, I] = max(ygt);
   %g2 = exp(-0.001*(k - k(I)).^2);
   %ygt_spec(:,j) = ygt .* g2;
   ygt_spec(:,j) = ygt;
end

figure(1)
pcolor(tau,ks,log(ygt_spec+1))
shading interp
set(gca,'ylim',[350, 1200],'Fontsize',16)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title('Floyd Guitar with Gaussian and Shannon filters');
```

Appendix C Music scale along with the frequency of each note in Hz

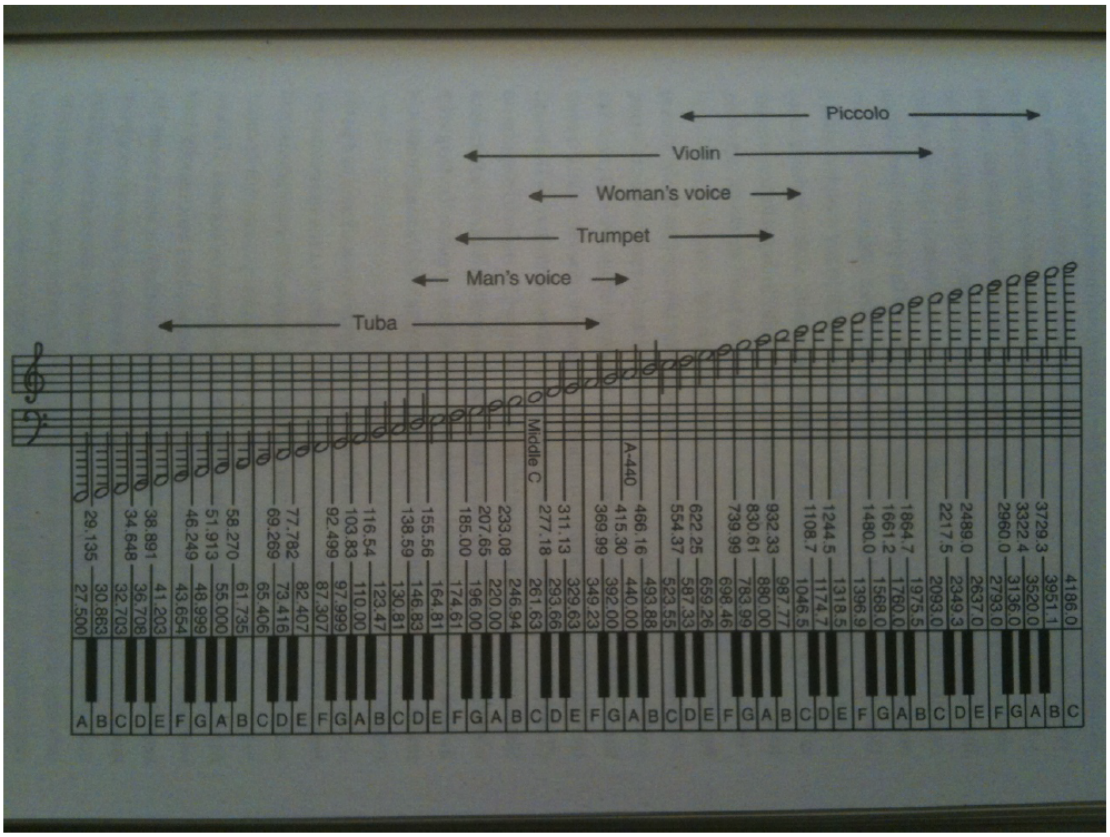


Figure 1: Music scale along with the frequency of each note in Hz